**Q1.**

문제 설명

작은 육각형으로 이루어진 큰 육각형 벌집이 있습니다. 벌집의 한 변을 구성하는 육각형의 개수를 나타내는 정수 n이 매개변수로 주어집니다. 한 변의 길이가 n인 육각형 벌집을 맨 위부터 시계방향으로 돌아가면서 달팽이처럼 채운 뒤, 위에서부터 내려오면서 좌에서 우로 벌집을 훑었을 때 나오는 숫자들을 차례대로 배열에 담아 return 하도록 solution 함수를 완성해주세요.

지도이(가) 표시된 사진

자동 생성된 설명

제한사항

1 ≤ n ≤ 500

입출력 예

n result

2 [1,6,2,7,5,3,4]

3 [1,12,2,11,13,3,18,14,10,19,4,17,15,9,16,5,8,6,7]

4 [1,18,2,17,19,3,16,30,20,4,29,31,21,15,36,32,5,28,37,22,14,35,33,6,27,34,23,13,26,24,7,12,25,8,11,9,10]

입출력 예 설명

입출력 예 #1

문제 예시와 같습니다.

입출력 예 #2

문제 예시와 같습니다.

입출력 예 #3

문제 예시와 같습니다.

**Q2.**

리눅스의 쉘 명령어가 하는 일을 유사하게 따라 하는 프로그램을 작성하려고 합니다.

다음 세 가지 명령어가 동작하도록 프로그램을 작성해주세요.

1.

이름 mkdir

사용 방법 mkdir directory

설명 특정 위치에 디렉토리를 생성합니다.

2.

이름 rm

사용 방법 rm directory

설명 특정 디렉토리를 삭제합니다. 삭제한 디렉토리의 하위 디렉토리들 또한 함께 삭제됩니다.

3.

이름 cp

사용 방법 cp source dest

설명 source 디렉토리를 dest로 복사합니다. source 디렉토리의 하위 디렉토리들 또한 복사되어야 합니다.

위 명령을 수행할 시스템에는 오직 디렉토리만 존재하며, 최상위 디렉토리는 "/"로 표시하고 그 하위 디렉토리는 "/root", "/usr", "/tmp", "/tmp/tmp2"와 같이 구분합니다.

제한사항

directory는 시스템의 디렉토리 구조를 나타내는 문자열 배열입니다.

directory의 길이는 1 이상 20 이하입니다.

directory의 원소는 각 디렉토리의 절대 경로(최상위 디렉토리부터의 경로)를 나타냅니다.

시스템상의 모든 디렉토리에 대한 경로가 주어집니다.

각 디렉토리 이름은 길이가 1 이상 10 이하입니다.

경로를 나타내는 문자열은 알파벳 소문자와 '/' 로만 이루어져 있습니다.

한 디렉토리 안에 같은 이름의 디렉토리가 두 개 이상 있는 경우는 없으며, 또한 잘못된 경로가 주어지는 경우는 없습니다.

디렉토리 경로는 사전 순으로 정렬되어 주어집니다.

command는 명령어들이 담긴 문자열 배열입니다.

command의 길이는 1 이상 20 이하입니다.

command의 원소는 쉘 명령어를 나타내는 문자열입니다.

문제에 주어진 명령어만 입력으로 주어지며, 잘못된 경로가 주어지는 경우는 없습니다.

경로는 항상 절대 경로(최상위 디렉토리부터의 경로)로만 주어집니다.

cp명령의 경우 자기 자신의 하위 디렉토리에 복사하는 명령은 주어지지 않습니다.

cp명령의 경우 복사하는 위치에 이름이 동일한 다른 디렉토리가 이미 있는 경우는 주어지지 않습니다.

"rm /" 명령은 입력으로 주어지지 않습니다.

rm 명령은 항상 디렉토리 하나를 정상적으로 삭제하는 경우만 입력으로 주어집니다. (존재하지 않거나 잘못된 경로, 없는 디렉토리 등을 삭제하는 경우는 없습니다)

mkdir 명령은 항상 디렉토리 하나를 정상적으로 생성하는 경우만 입력으로 주어집니다. (존재하지 않거나 잘못된 경로, 해당 경로에 같은 이름의 디렉토리가 이미 있는 경우, 중간 경로에 해당하는 디렉토리가 없는 경우 등은 입력으로 주어지지 않습니다.)

모든 명령어를 수행한 후 디렉토리 구조를 문자열 형태로 배열에 담아 사전 순으로 정렬해서 return해주세요.

입력되는 디렉토리 구조와 마찬가지로, 모든 디렉토리에 대한 절대 경로를 문자열 형태로 담은 뒤 사전 순으로 정렬해서 return 하면 됩니다.

'/'가 알파벳보다 사전 순으로 더 앞에 온다고 가정합니다.

입출력 예 #1

directory

[

"/",

"/hello",

"/hello/tmp",

"/root",

"/root/abcd",

"/root/abcd/etc",

"/root/abcd/hello"

]

command

[

"mkdir /root/tmp",

"cp /hello /root/tmp",

"rm /hello"

]

result

[

"/",

"/root",

"/root/abcd",

"/root/abcd/etc",

"/root/abcd/hello",

"/root/tmp",

"/root/tmp/hello",

"/root/tmp/hello/tmp"

]

입출력 예 #2

directory

[

"/"

]

command

[

"mkdir /a",

"mkdir /a/b",

"mkdir /a/b/c",

"cp /a/b /",

"rm /a/b/c"

]

result

[

"/",

"/a",

"/a/b",

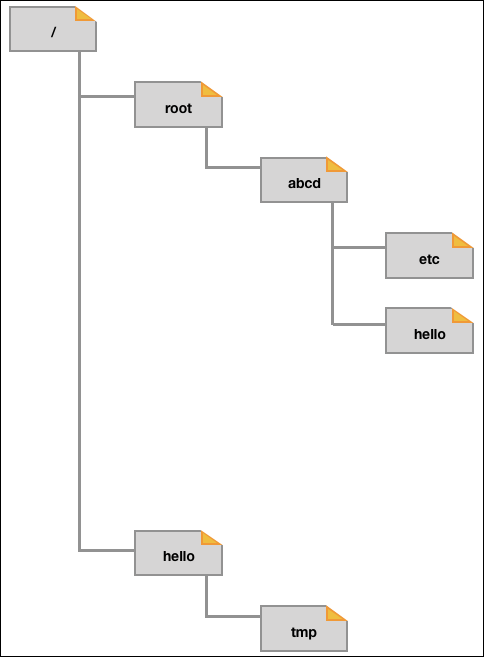
"/b",

"/b/c"

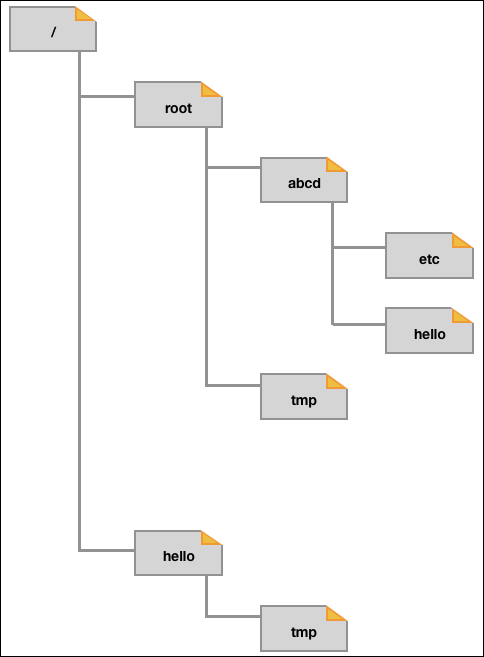
]

입출력 예 설명

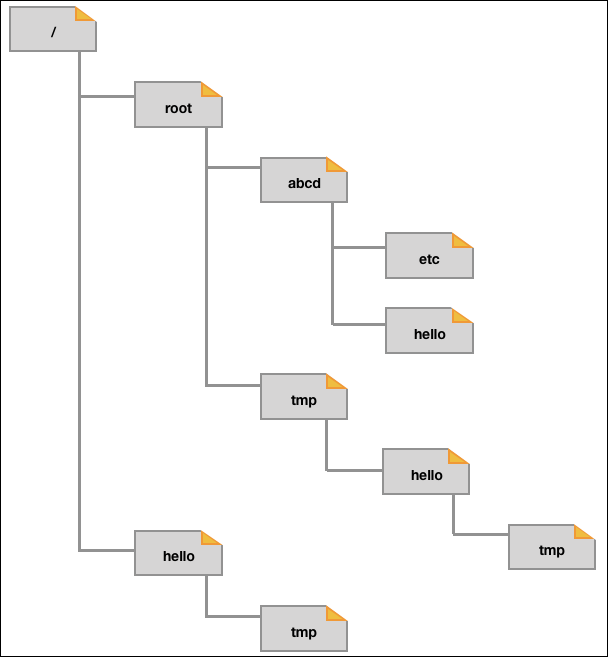
입력으로 주어진 초기 디렉토리 구성은 다음과 같습니다.



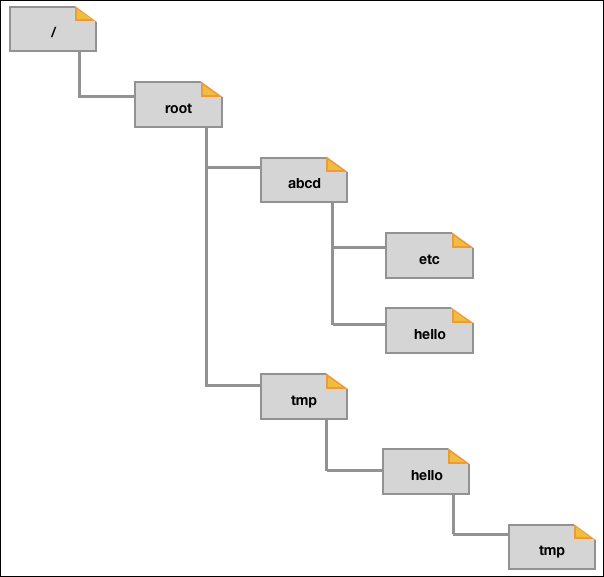
"mkdir /root/tmp" 명령을 수행한 후 디렉토리 구성은 다음과 같습니다.



"cp /hello /root/tmp" 명령을 수행한 후 디렉토리 구성은 다음과 같습니다.

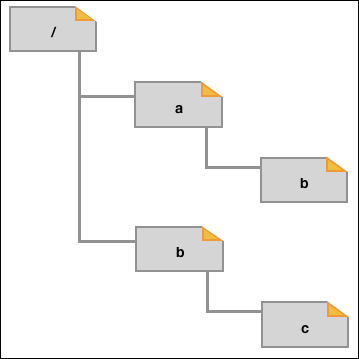


"rm /hello" 명령을 수행한 후 최종 디렉토리 구성은 다음과 같습니다.



입출력 예 #2

명령을 모두 처리한 후 디렉토리 구성은 다음과 같습니다.



123456

class Solution {

public String[] solution(String[] directory, String[] command) {

String[] answer = {};

return answer;

}

}

**Q3.**

문제 설명

발전기 여러 대가 다양한 길이의 단방향 전선으로 연결되어 있습니다. 가동 중인 발전기는 자기 자신으로부터 나가는 전선을 통해 다른 발전기에 전력을 공급합니다. 단, 발전기마다 수명이 정해져 있으며, 수명이 다한 발전기는 가동을 중단합니다. 처음에는 모든 발전기가 가동 중입니다.

발전기가 계속 가동 중이려면 다음 두 조건을 모두 만족해야 합니다.

발전기의 남은 수명이 0보다 커야 합니다. 수명은 하루에 1씩 감소합니다.

예를 들어, 어떤 발전기의 남은 수명이 3이라면 이 발전기는 3일 뒤 가동을 중단합니다.

해당 발전기로 들어오는 전선 중 적어도 1개 이상의 전선에서 전력이 공급되어야 합니다.

이때, 모든 전력은 하루에 길이 1만큼 이동합니다. 예를 들어, 발전기 A에서 발전기 B까지 길이 3짜리 전선으로 연결되어 있다면, 발전기 A가 가동을 중단한 후 정확히 3일 뒤 해당 전선의 전력 공급이 끊어집니다.

발전기들의 수명 ages와 각 전선에 대한 정보 wires가 매개변수로 주어집니다. 시간이 지나 모든 발전기의 가동이 중단된 후, 가동이 중단된 순서대로(동시에 중단된다면 작은 번호 먼저) 발전기들의 번호를 배열에 담아 return 하도록 solution 함수를 완성해주세요.

제한 사항

ages의 길이는 1 이상 100,000 이하입니다.

발전기들의 번호는 1, 2, 3, ..., (ages의 길이)까지입니다.

ages[x]는 x+1번 발전기의 수명을 의미합니다.

ages에 있는 모든 값은 1 이상 1015 이하인 자연수입니다.

wires의 세로(행) 길이는 (ages의 길이) 이상 300,000 이하입니다.

wires의 각 행은 [start, end, length] 형태입니다.

각 행은 start번 발전기에서 end번 발전기까지 length 길이만큼의 전선으로 연결되어 있음을 의미합니다.

1 ≤ start ≤ (ages의 길이), 1 ≤ end ≤ (ages의 길이), start ≠ end입니다.

1 ≤ length ≤ 1,000,000,000입니다.

모든 발전기는 최소 1개 이상의 다른 발전기로부터 출발하는 전선이 연결되어 있습니다.

임의의 두 행은 start, end 값 중 최소 1개 이상의 값이 다릅니다. 즉, start번 발전기에서 end번 발전기로 전력을 보내는 전선은 최대 1개입니다.

입출력 예

ages wires result

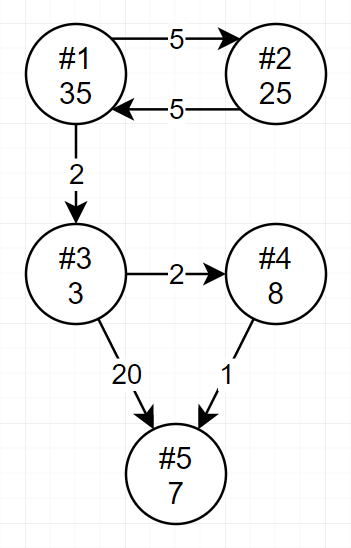
[35,25,3,8,7] [[1,2,5],[2,1,5],[1,3,2],[3,4,2],[3,5,20],[4,5,1]] [3,4,5,2,1]

[8,13,5,8] [[1,3,10],[3,4,1],[4,2,2],[2,1,3]] [3,4,1,2]

입출력 예 설명

입출력 예 #1

주어진 입력을 그림으로 표현하면 다음과 같습니다.



발전기가 가동이 중지되는 과정을 표로 나타내면 다음과 같습니다.

시간 내용

처음 모든 발전기가 가동 중입니다.

3일 후 3번 발전기의 수명이 다하여 가동을 중단합니다.

5일 후 4번 발전기가 유일하게 전력을 공급하던 3번 발전기로부터 전력 공급이 끊겨 가동을 중단합니다.

6일 후 5번 발전기가 더 이상 4번 발전기로부터 전력을 공급받지 못합니다. 하지만 아직 3번 발전기로부터 연결된 전선의 길이가 길어, 처음으로부터 23일 후가 되기 전까지는 전력을 계속 공급받을 수 있습니다.

7일 후 5번 발전기의 수명이 다하여 가동을 중단합니다.

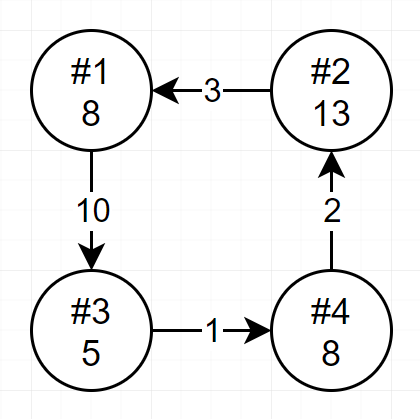
25일 후 2번 발전기의 수명이 다하여 가동을 중단합니다.

30일 후 1번 발전기가 더 이상 2번 발전기로부터 전력을 공급받지 못하여 가동을 중단합니다.

따라서 [3,4,5,2,1] 을 return 해야 합니다.

입출력 예 #2

주어진 입력을 그림으로 표현하면 다음과 같습니다.



발전기가 가동이 중단되어가는 과정을 표로 나타내면 다음과 같습니다.

시간 내용

처음 모든 발전기가 가동 중입니다.

5일 후 3번 발전기의 수명이 다하여 가동을 중단합니다.

6일 후 4번 발전기가 더 이상 3번 발전기로부터 전력을 공급받지 못하여 가동을 중단합니다.

8일 후 1번 발전기의 수명이 다하여 가동을 중단합니다. 한편, 2번 발전기도 더 이상 4번 발전기로부터 전력을 공급받지 못하여 가동을 중단합니다.

1번 발전기와 2번 발전기는 동시에 가동을 중단하므로, 번호가 더 작은 1번을 먼저 배열에 담아야 합니다.

따라서 [3,4,1,2] 을 return 해야 합니다.

12345678910111213

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

// ages\_len은 배열 ages의 길이입니다.

// wires\_row\_len은 2차원 배열 wires의 행(세로) 길이입니다.

// wires\_col\_len은 2차원 배열 wires의 열(가로) 길이입니다.

// wires[i][j]는 wires의 i번째 행의 j번째 열에 저장된 값을 의미합니다.

int\* solution(long long ages[], size\_t ages\_len, int \*\*wires, size\_t wires\_row\_len, size\_t wires\_col\_len) {

// return 값은 malloc 등 동적 할당을 사용해주세요. 할당 길이는 상황에 맞게 변경해주세요.

실행 결과

실행 결과가 여기에 표시됩니다.